通过spark-redshift工具包读取Redshift上的表

Spark Data Source API是从Spark 1.2开始提供的，它提供了可插拔的机制来和各种结构化数据进行整合。Spark用户可以从多种数据源读取数据，比如Hive table、JSON文件、Parquet文件等等。我们也可以到http://spark-packages.org/（这个网站貌似现在不可以访问了）网站查看Spark支持的第三方数据源工具包。本文将介绍新的Spark数据源包，通过它我们可以访问 Amazon Redshift Service，这个工具包叫做spark-redshift。spark-redshift主要由Databricks维护，并且有SwiftKey等公司贡献代码。

在spark-redshift诞生之前，Spark的JDBC库是唯一可以访问Redshift的方式。这种方式在查询非常少的数据（比如100条）的情况下是非常适合的，但是如果用于处理大规模数据，将会变得非常慢！因为JDBC提供了基于ResultSet方式，而这种方式在一个线程中接收所有的数据。此外，如果使用JDBC在Redshift中存储大规模的数据集，唯一可行的方式就是在Redshift中同一个数据库中进行。基于JDBC的INSERT/UPDATE查询也仅仅适合小规模的数据。对于那些想在Redshift中加载或者存储大规模数据的用户来说，JDBC实在是有太多的性能和吞吐量问题亟待改变。

然而使用spark-redshift可以简化和Redshift整合的步骤，使得我们可以从Redshift中加载或者存储大规模的数据。为了了解它是怎么工作的，让我们来看看如何用Redshift数据库来和其他的数据源的数据集进行集成。

我们还将在本文探讨spark-redshift是如何扩展的。一般情况下，数据需要从HDFS上移到Redshift中进行分析。然而spark-redshift将允许Redshift无缝地操作（通过统一的数据源API）存储在S3、Hive tables、CSV或者存储在HDFS上的Parquet文件。这将使得ETL工作变得简单，用户只需要将注意力放在业务逻辑上，而且这提供了统一的系统视图。

从Redshift中读取数据

假如你需要通过Spark读取Redshift表中的所有数据，并且和来自其他数据源的数据进行整合，比如Hive，从Redshift的表中加载数据并转换成DataFrame实例的命令可以这么写：

```
//////////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2015-10-21
 Time: 23:59
 bolg:
 本文地址：/archives/1517
 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 过往记忆博客微信公共帐号：iteblog_hadoop
//////////////////////////////////////////////////////////////////////////
val jdbcURL =  """jdbc:redshift://www.iteblog.com:5439/
                testredshift?user=iteblog&amp;amp;
               password= W9P3GC42GJYFpGxBitxPszAc8iZFW"""
```

```
val tempS3Dir = "s3n://spark-redshift-testing/temp/"
val salesDF = sqlContext.read
   .format("com.databricks.spark.redshift")
   .option("url", jdbcURL) //Provide the JDBC URL
   .option("tempdir", tempS3Dir) //User provides a temporary S3 folder
   .option("dbtable", "sales") //or use .option("query",”select * from sales”)
   .load()
```

上面的命令为 Redshift 表提供了DataFrame 实例，用户仅仅需要提供JDBC URL，用于缓存Redshift表数据的S3上面的临时文件夹以及需要查询表的名字。

在Spark中DataFrame实例可以注册成一个临时表，然后我们就可以对它进行查询

```
salesDF.registerTempTable("sales_from_redshift")
val newSalesDF = sqlContext.sql("SELECT count(*) FROM sales_from_redshift")
```

我们可以通过SQL命令行接口实现同样的结果

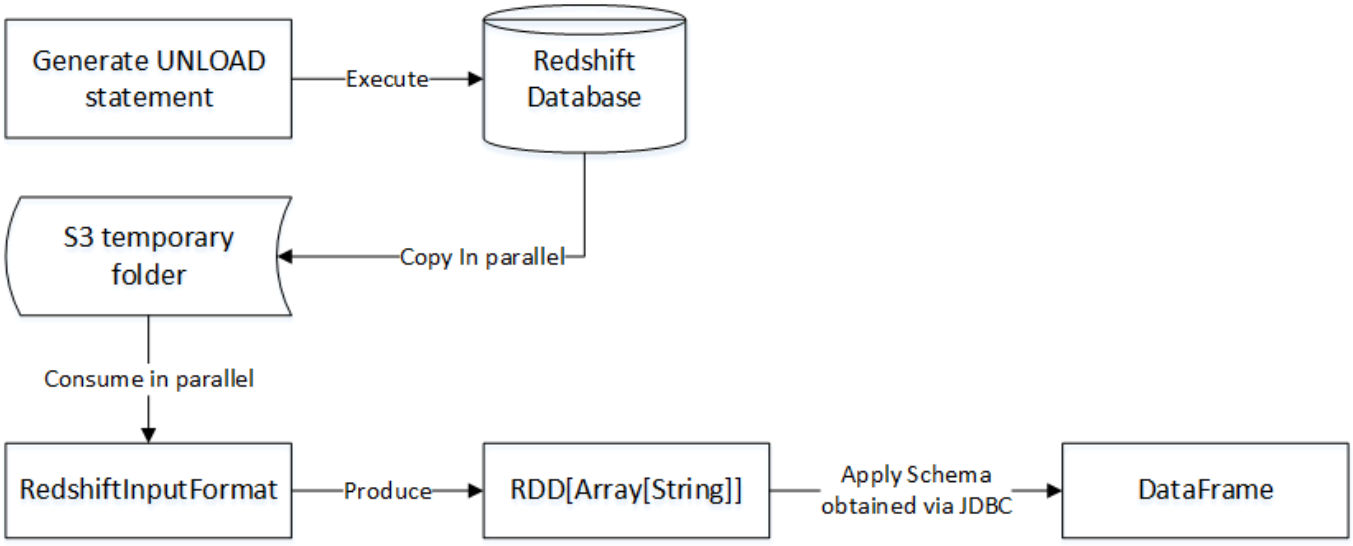
```
//////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2015-10-21
 Time: 23:59
 bolg:
 本文地址：/archives/1517
 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 过往记忆博客微信公共帐号：iteblog_hadoop
//////////////////////////////////////////////////////////////////////
CREATE TEMPORARY TABLE sales_from_redshift
USING com.databricks.spark.redshift
OPTIONS (
dbtable 'sales',
tempdir 's3n://spark-redshift-testing/temp/',
url 'jdbc:redshift://www.iteblog.com:5439/
testredshift?user=redshift&amp;amp;password=W9P3GC42GJYFpGxQtaCBitxPszAc8iZFW');

SELECT count(*) FROM sales_from_redshift;
```

在Spark中，我们将检索的Redshift表注册成一个名为sales_from_redshift的临时表，我们可以直接在这个名字的表上进行查询：

```
SELECT count(*) FROM sales_from_redshift;
```

在这个语句的背后，spark-redshift执行Redshift的UNLOAD命令来并行地复制Redshift上表的数据，并缓存在用户指定的S3文件夹中。然后使用Hadoop InputFormat API来并行地读取这些存储在S3上面的文件，并将它映射成RDD实例。最后它利用JDBC元数据检索功能来将被检索表的模式生成DataFrame实例。如下图所示：



spark-redshift工具包并不能自动删除存储在S3上面的文件。所以建议使用S3上专门用于临时存放文件的地方来存放文件，使得这些临时文件可以在指定时间后自动删除。

将数据写到Redshift

Spark Data Sources API是一个强大的ETL工具。在大数据系统中比较常见耳朵应用场景是从一个海量数据系统读取数据，并在这些数据上分布式地进行一些transformations操作，然后将这些计算好的数据存储到其他系统中。比如我们经常会从Hive中读取数据，然后将表中的数据复制到Redshift中，以便允许我们进行交互式地处理。spark-redshift工具包特别适合这种应用场景。

假如我们可以直接在Spark环境中读取Hive中表的数据，然后我们需要将这些数据拷贝到Redshift相应的redshift_transaction表中，我们可以进行如下操作：

```
//////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2015-10-21
 Time: 23:59
 bolg:
```

本文地址：/archives/1517
过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
过往记忆博客微信公共帐号：iteblog_hadoop
//

```
sqlContext.sql("SELECT * FROM transaction")
        .write.format("com.databricks.spark.redshift")
        .option("url", jdbcURL)
        .option("tempdir", tempS3Dir)
        .option("dbtable", "redshift_transaction")
        .mode(SaveMode.Overwrite)
        .save()
```

使用SQL CLI也可以达到相同的目的，如下：

```
CREATE TABLE redshift_transaction
USING com.databricks.spark.redshift
OPTIONS (
dbtable 'redshift_transaction',
tempdir 's3n://spark-redshift-testing/temp/',
url 'jdbc:redshift://www.iteblog.com:5439/
testredshift?user=redshift&amp;amp;password=W9P3GC42GJYFpGxQtaCBitxPszAc8iZFW')
AS SELECT * FROM transaction;
```

注意上面Scala代码中的 mode(SaveMode.Overwrite)，这暗示spark-redshift在该表存在的时候去覆盖。默认情况下（也就是SaveMode.ErrorIfExists），如果需要创建的表存在，那么spark-redshift将会抛出一个异常。SaveMode.Append模式的意思就是如果表不存在则创建；如果表存在，则将数据追加到表的后面。最后一种模式是SaveMode.Ignore，在这种模式下，如果表不存在，则创建表；如果表存在，后面的整个命令就不会执行。

在写数据的时候，spark-redshift工具包将使用JDBC首先在Redshift 中创建表，然后它将分区好的RDD复制到S3临时目录中；最后，它允许Redshift的COPY命令分布式的来将S3上保存的数据复制到Redshift相应的表中。如下图所述：

和其他数据源整合

通过spark-redshift工具包读取的数据将自动地转换成DataFrame对象。Redshift用户可以将存储在S3上的Redshift表和H存储在HDFS上的Hive表、CSV或者Parquet进行Join操作。这个特性对用户来说是非常重要的。

本文翻译自：https://databricks.com/blog/2015/10/19/introducing-the-spark-redshift-package.html